THE UNIVERSITY *of* EDINBURGH

Planning How to Grasp Objects in a Cluttered Environment

Michael Paul Wingham

Master of Philosophy

University of Edinburgh

1977

ABSTRACT

This thesis deals with the problem of finding trajectories of objects through space which do not result in collisions. Specifically, a method is presented which calculates such trajectories in the case of a robot attempting to grasp a certain body in a cluttered environment, i.e. where there are other bodies nearby. We restrict ourselves to bodies with planar faces, robots with certain physical characteristics, and a specific class of trajectories.

We present a method of describing bodies in terms of the shape of their faces. The robot's trajectories are not predetermined, but instead are calculated on the basis of the present configuration of bodies in the robot's world. Once the body to be grasped has been chosen, we examine all nearby objects to determine the region of space which is inaccessible to the robot. This region is projected onto a suitable plane, and making use of algorithms to compute the intersection and union of two-dimensional figures, we are able to find a region of the surface of the body to be grasped which is sufficiently distant from neighbouring objects, assuming such a region exists. This region in turn allows us to calculate a set of feasible robot trajectories for grasping the body. We also suggest a possible approach to computing more general trajectories involving both the robot and a body.

CONTENTS

1.0 INTRODUCTION

Research in the field of automatic assembly can be roughly divided into two areas. Firstly, there is "high level" planning which deals with the problems of producing a satisfactory ordering of assembly operations to obtain a desired configuration of components. Secondly is trajectory planning which concerns itself with the actual execution of basic assembly operations in accordance with the restriction that no two bodies can occupy the same region of space at the same time. To date, much has been done in the former area; for example [Fikes and Nilsson, 1971], [Sussman, 1973], [Sacerdoti, 1975]. Very little work has been done in trajectory planning, however. This is partly due to the belief that trajectory planning is highly dependent upon the geometry of the individual robot and is hence not readily extendable to other systems. Also, most approaches to the problem tend to be computationally unwieldy. It is clear, however, that in any implementation of a computer-controlled automatic assembly system, some sort of trajectory planning is necessary.

One aspect of this area is the problem of grasping an object with a robot arm. In the TC-COPY system [Winston, 1971] which dealt with duplicating blocks world structures using blocks in a spare parts warehouse area, this problem was avoided by arranging structures in such a way that no collisions could occur during the grasping process. Blocks would be grasped by one of the two pairs of opposite vertical faces. A pair of faces would not be used for

grasping if their separation was greater than the maximum jaw opening of the robot hand. Otherwise, the block would be grasped by the pair of faces which was most closely aligned with back to front; since structures were arranged from left to right, this reduced the chance of a collision. The method used in AL [Finkel et al., 1974] is to assign a grasping point and approach trajectory to each body in the robot's world. Again it is assumed that throughout the duration of the task the given trajectory will not result in a collision. Neither system deals with the problem of moving the object through space once it has been grasped.

In the Versatile Assembly Program [Ambler et al., 1975] a heap of objects was viewed by a television camera mounted vertically above it. The aim was to separate and identify objects in the heap for later use in an assembly. From the television picture, it was possible to identify horizontal projections from the heap, which the robot was then directed to grasp. This system was more versatile than the two mentioned previously, however it required a visual image projected onto a plane which is perpendicular to the trajectory of the grasping arms, i.e. the camera had to be mounted vertically above the heap.

This paper considers a more robust method for computing trajectories to grasp objects. The trajectories are not predetermined, but instead are calculated on the basis of the present configuration of bodies in the robot's world. This method is made computationally feasible by reducing what is a 4-dimensional

problem (i.e. time and 3-space) to a 2-dimensional one. The time dimension is eliminated by looking at the volume swept by an object over an interval of time. One space dimension is eliminated by projecting all 3-D information onto a suitable plane. This method also suggests a possible approach to computing more general trajectories involving both the robot and an object.

2.0 RESTRICTIONS

For the purposes of this paper, the following restrictions are imposed on the robot and the objects in its world. All bodies must be rigid and have planar faces. Edges can be either convex or concave. The robot must have two opposing faces which come in contact with the surface of the body to be grasped. Each of these two faces will be referred to as a "hand". We assume the hands are parallel, and that the projection of one onto the other is coincident with it. Each hand has a rigid attachment to the remainder of the robot, which we shall call the "arm".

In order to eliminate the time dimension from our calculations of possible trajectories for grasping objects, we assume that the volume of space swept by each hand and arm in each possible trajectory can be described as a subvolume of some cuboid. For example, suppose we wish to grasp a block. The obvious sort of trajectory to use is to move the hands to a position above the block so that they are parallel to a pair of opposite faces of the block, and the separation of the hands is slightly greater than the separation of the faces. The hands are vertically lowered to some point below the top edge of the block and then moved toward the surface of the block until contact is established. As can be seen in diagram 1, the trajectory of each hand and arm can be enclosed in a cuboid, and we assume that the position of this pair of cuboids is the only thing relevant to avoiding a collision, i.e. the space occupancy of the remainder of the robot is neglected. Furthermore,

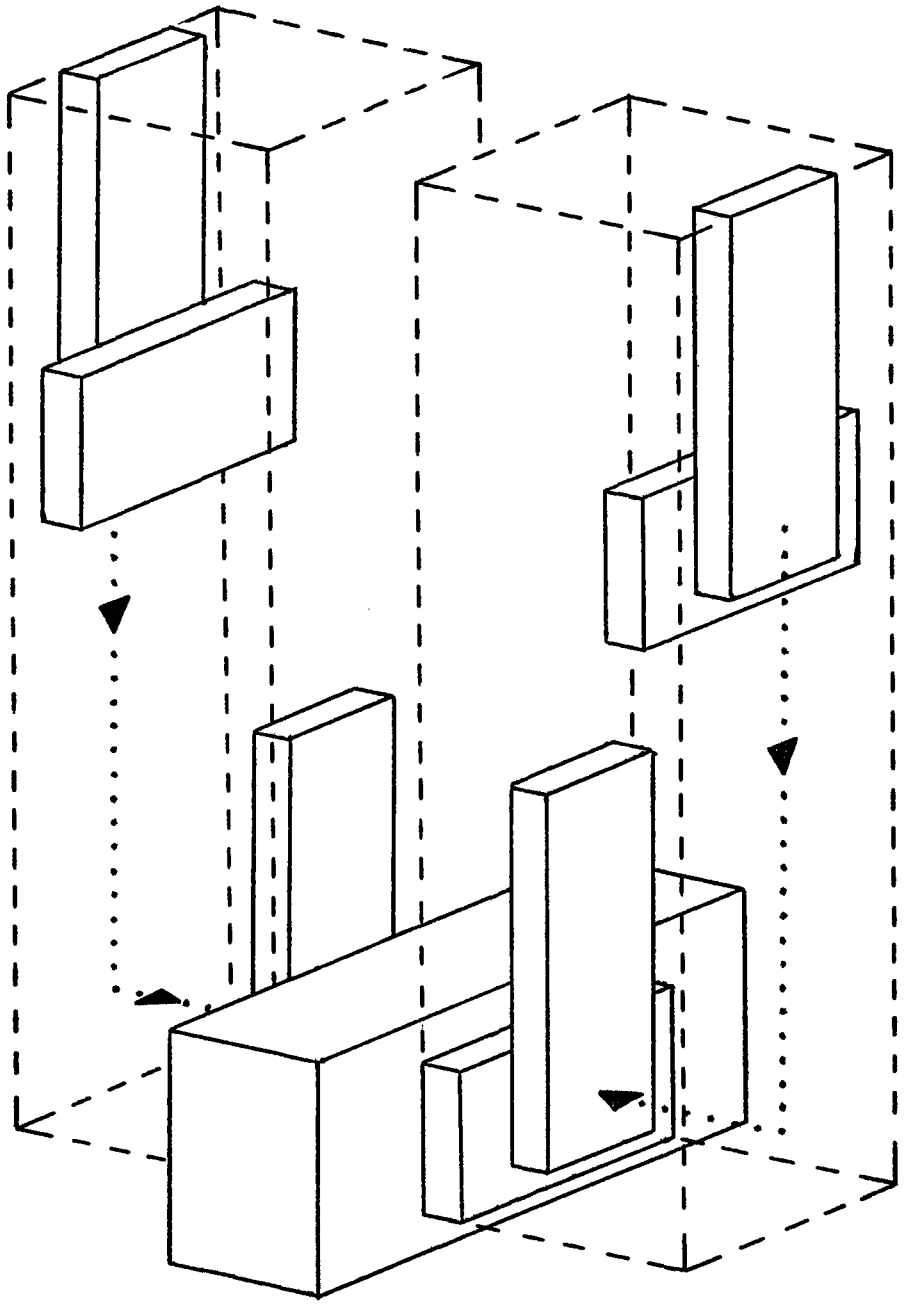


Diagram 1. This diagram shows the trajectory, indicated by the dotted lines, of the robot hands in the process of grasping a block. The volume swept out during this trajectory can be enclosed in a pair of cuboids, illustrated here by the dashed lines.

we grasp objects only at opposite faces, i.e. we don't consider the possibility of grasping a body by a face and an edge or corner.

Thus in terms of obstacle avoidance, the following variables must still be determined :

a. Which pair of faces should the object be grasped by ?

b. At what point along the length of the face should the hands be positioned ?

c. How far down should the hands be lowered ?

The solution to these problems will be dependent upon the shape of the body to be grasped and the position of nearby objects. It is the purpose of this paper to present such a solution.

3.0 REPRESENTATION OF BODIES

In order to perform any sort of manipulation, we must know the shape and position of every body in the robot's world. The following representation is employed. Each body has an associated set of axes referred to as the "local" or "body" axes. The origin specifies the location of the body, while the directions of the axes specify the orientation. Location and orientation together give the position of a body. We also have a system of world coordinates which remain fixed throughout. It is convenient to think of the position of a body as being the transformation required to map the coordinates of a point from body to world coordinates. We can express these transformations by making use of a system of homogeneous coordinates [Ambler and Popplestone, 1975]. Let the vector S = [a b c] be the coordinates of the origin of the body axes with respect to world coordinates. Let the matrix

$$R = \begin{bmatrix} u1 & v1 & w1 \\ u2 & v2 & w2 \\ u3 & v3 & w3 \end{bmatrix}$$

be the direction matrix of the body axes, where [u1 v1 w1], [u2 v2 w2], and [u3 v3 w3] are the direction cosines of the body x, y, and z axes respectively. If we represent points in 3-space by vectors of the form [x y z 1], then the matrix

$$T = \left[\begin{array}{c|c} \multirow{2}{*}{R} & \begin{matrix}0\\0\end{matrix} \\ & 0 \\ \hline S & 1 \end{array}\right] = \begin{bmatrix} u1 & v1 & w1 & 0 \\ u2 & v2 & w2 & 0 \\ u3 & v3 & w3 & 0 \\ a & b & c & 1 \end{bmatrix}$$

maps points in 3-space from local to world coordinates. The inverse mapping is

$$T^{-1} = \left[\begin{array}{c|c} \multirow{2}{*}{R'} & \begin{matrix}0\\0\end{matrix} \\ & 0 \\ \hline -SR' & 1 \end{array}\right]$$

where R' is the transpose of R. If we represent a plane $Ax+By+Cz+D=0$ by a column vector of the form

$$L = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$$

then T also maps planes from local to world coordinates.

Since all interactions between bodies occur at their surfaces, it is desirable to describe a body in terms of the shapes and relative positions of its faces. (Thus we do not use a method of description similar to Braid [Braid, 1975] in which bodies are represented in terms of syntheses of primitive solids.) Each face has an associated set of axes with the x and y axes lying in the face, and the z-axis normal to and pointing outward from the face. The position of the axes with respect to body coordinates is represented by a 4x4 matrix as above.

We are now left with the problem of representing the shape of a face. Since we have restricted ourselves to bodies with planar faces, we are essentially dealing with the problem of representing "figures" in a plane. The definition of a figure will be given in the next section, but in the case of a face with no holes in it, its figure is simply a polygon.

3.1 Definition And Representation Of Figures

If we were to require that faces of bodies have no holes in them, then faces could simply be represented as polygons. We must extend this representation, however, in order to include the possibility of holes. For this purpose, the notion of a "figure" is introduced. A figure is a finite set of simple closed directed curves, each of these curves consisting of a finite number of straight line segments. These curves cannot cross each other, but may coincide at a finite number of points. There is also a restriction on the possible directions of the curves as follows. Let C1 and C2 be any two distinct curves of a figure F. If it is possible to join C1 and C2 by a directed curve D which does not cross any other curve of F, then C1 and C2 must cross D from different directions. Examples are given in diagram 2.

Each straight line segment is referred to as an "edge", and the endpoints as "vertices". Edges have directions, which are identical to the direction of the curve to which they belong. Suppose an edge has vertices (a1, b1) and (a2, b2), and direction vector [a2-a1, b2-b1]. Let P=(x,y) be any point. We say that P is to the "left" or "right" of the edge if the determinant of the matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ a1 & a2 & x \\ b1 & b2 & y \end{bmatrix}$$

is positive or negative respectively. This definition coincides with one's intuitive notion of left and right.

A figure divides the plane into a number of connected open regions, and these regions can be partitioned into two classes. In each region there exist points which have a unique closest edge. If such points are to the left of the edge, then the region to which they belong is "inside"; otherwise it is "outside". Thus any point in the plane is either on the figure, in an inside region, or in an outside region. For convenience, we will refer to a point P being in an inside or outside region of a figure F, by saying that P is inside or outside F respectively.

We say that a vertex V2 is the successor of a vertex V1, if V1 and V2 are connected by an edge E, and the direction of the vector V2-V1 coincides with the direction of the edge. We also say that V1 "initiates" E and V2 "terminates" E.

Each curve of a figure can be expressed as a permutation of some set of points in the plane, each point corresponding to a vertex of the curve. Each point (vertex) in the domain of the permutation is mapped to its successor point (vertex). A figure is represented as a set of permutations. See diagram 3 for examples.

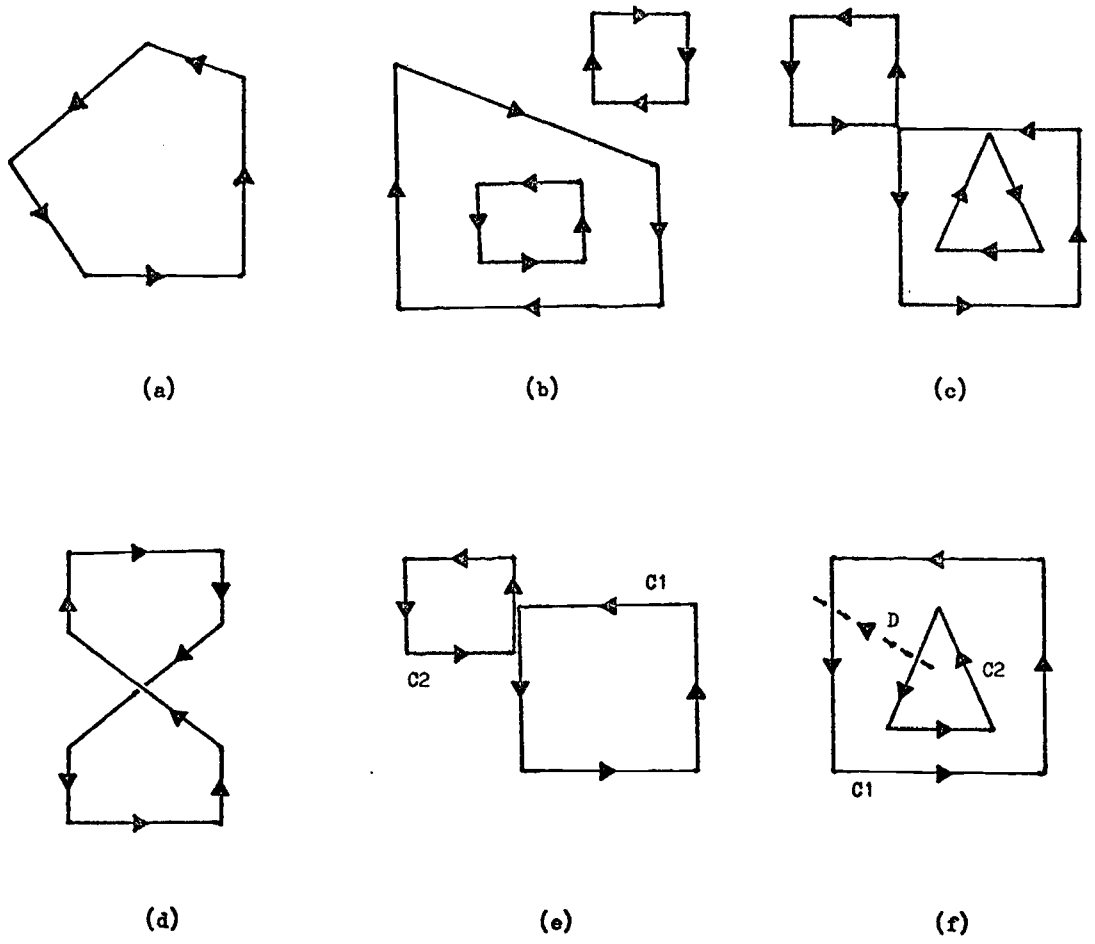


Diagram 2. Examples of figures are given in (a) to (c); the arrows denote the direction of the curves. (d) to (f) are examples of invalid figures. In (d), the curve crosses itself. In (e), curves C1 and C2 coincide at an infinite number of points. In (f), curves C1 and C2 can be joined by a directed curve D (shown by the dashed line) which does not cross any other curve, but C1 and C2 cross D from the same direction.

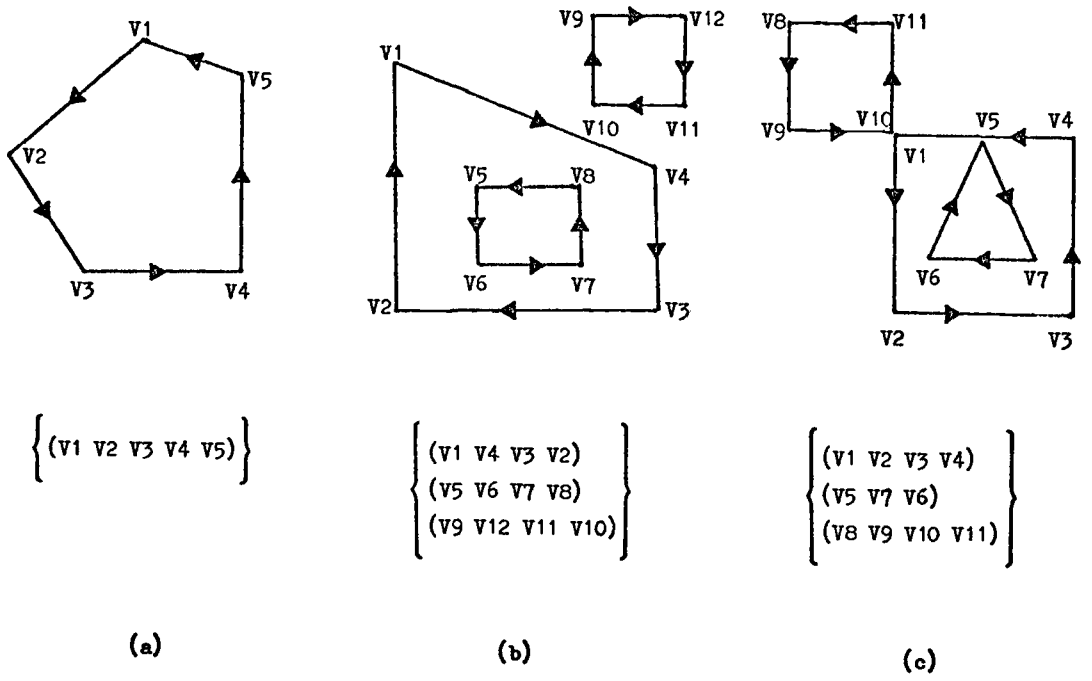


Diagram 3. In (a) to (c), the figures from Diagram 2.(a)-(c) have been reproduced, and their representations as sets of permutations have been shown. In (c), note that although V1 and V10 coincide, they belong to different curves and hence are distinct.

4.0 OPERATIONS ON FIGURES

We now discuss two binary operations which can be defined on figures, namely intersection and union, which we shall denote by the symbols "." and "+" respectively. These should be distinguished from the standard set operations of intersection and union which will be denoted by "n" and "u" respectively. Let "<=" be the symbol for set inclusion. For any figure F, let I(F) and O(F) denote the sets of points belonging to the inside and outside regions of F respectively. Let F and G be any two figures. Then H=F.G is defined as the figure H such that

$$I(F) \cap I(G) = I(H)$$

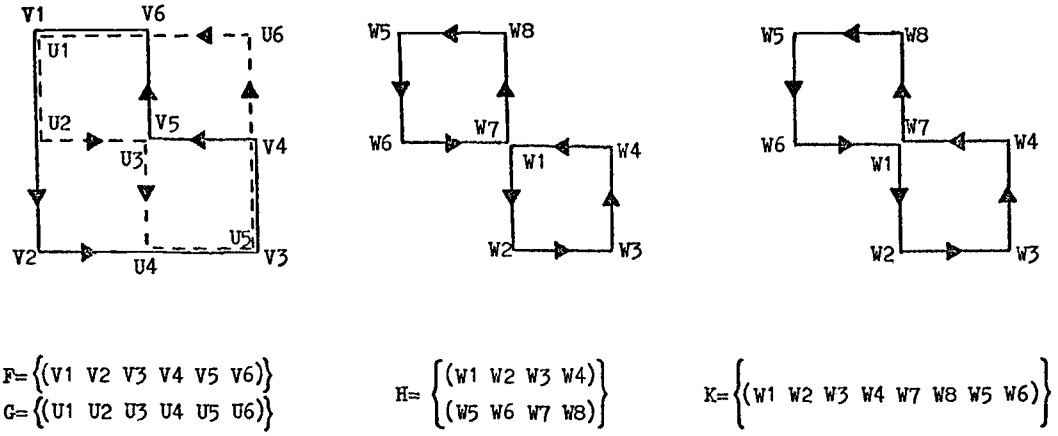
and $$O(F) \cup O(G) \subseteq O(H) \subseteq (O(F) \cup O(G)) \cup (F \cup G)$$

Similarly, K=F+G is the figure K such that

$$I(F) \cup I(G) \subseteq I(K) \subseteq (I(F) \cup I(G)) \cup (F \cup G)$$

and $$O(F) \cap O(G) = O(K)$$

These two operations are not strictly well-defined, in that H and K are not always uniquely determined; see diagram 4 for example. However, we regard any two figures which have the same inside and outside regions as being equivalent.



$F=\{(V1\ V2\ V3\ V4\ V5\ V6)\}$
$G=\{(U1\ U2\ U3\ U4\ U5\ U6)\}$

$H=\{(W1\ W2\ W3\ W4)\ (W5\ W6\ W7\ W8)\}$

$K=\{(W1\ W2\ W3\ W4\ W7\ W8\ W5\ W6)\}$

(a) (b) (c)

Diagram 4. In (a), figures F and G are represented by the solid and dashed lines respectively. The intersection of F and G can be represented by the figure H as in (b), or by K as in (c).

4.1 Determining Whether A Point Is Inside, Outside, Or On A Figure

We define the distance of an edge E to a point P as the shortest distance from P to any point on E. Suppose we wish to determine whether a point P lies on, inside, or outside a figure F. We do this by finding a closest edge E to P, and all the closest vertices V1, V2,...Vn to P. Let the distance from E to P be d1, and the distance from the Vi's to P be d2. If d1 or d2 equals zero, then P lies on F. If d1 and d2 are both greater than zero, then either d1 is less than d2, or d1 is equal to d2. In the former case, P is inside F if and only if P lies to the left of E. The latter case is more complex. If there exists a closest vertex Vj, which does not coincide with any other vertex of F, then P is inside F if and only if the interior angle at Vj is concave (greater than 180 degrees). Otherwise, suppose we have k closest vertices V1, V2,...Vk which coincide at some point Q. We examine all the edges of F which have Q as an endpoint, and find an edge E such that the angle between E and the line QP is minimum. Suppose the endpoint of E which lies on Q corresponds to the vertex Vi. Then P is inside F if and only if the interior angle at Vi is concave. See diagram 5 for examples.

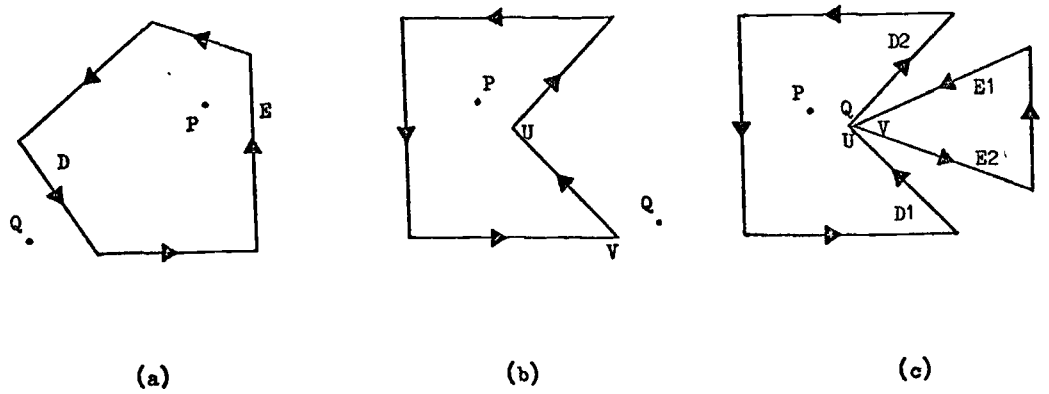


Diagram 5. In (a), P is closest to the edge E and lies to the left of it; hence P is inside the figure. On the other hand, Q is closest to D but lies to the right of it; hence Q is outside the figure.

In (b), P is closest to the vertex U. Since the angle at U is concave, P is inside the figure. Q is closest to the vertex V which is convex, hence Q is outside the figure.

In (c), P is closest to two vertices, U and V, which coincide at the point Q. Edges D1 and D2 have Q as an endpoint, as do edges E1 and E2. However, edge D2 makes the minimum angle with line QP. Since D2 corresponds to the vertex U, and the angle at U is concave, P must be inside the figure.

4.2 Algorithms For Intersection And Union Of Figures

We will now describe an algorithm for computing the intersection of two figures. An algorithm for union also exists but is not described, since it is quite similar to the one for intersection.

We say that two edges, D and E, cross each other if there exist two points in D which lie on either side of E, and two points in E which lie on either side of D. This implies that D and E have a point in common which will be referred to as the "crossing point".

Suppose that D and E are two successive edges of some figure F, and that D and E are joined at the vertex V. Let D′ and E′ be the two half lines emanating from V which coincide with D and E respectively. We say that a point P is inside the angle formed by D and E at V if either P lies on D′ or the clockwise angle between D′ and the line VP is less than the clockwise angle between D′ and E′. (see diagram 6)

One further definition is that of a "critical point". A critical point of a pair of figures (F,G) is any point in the plane which is either a vertex in F or G, or a crossing point of an edge in F and an edge in G. Diagram 7 gives an example of a pair (F,G) and its critical points. To each critical point P we associate a set S which consists of all the vertices which lie on P and all the edges which pass through P. Critical points are relevant in that

the vertices of the intersection H of two figures F and G, will be a subset of the set of critical points of the pair (F,G). It is necessary to determine which critical points correspond to vertices in H, and which edges are initiated or terminated by these vertices. We intend to construct a set of ordered triples of the form

(<initial edge>, <vertex>, <final edge>)

where <vertex> is a vertex V of H
<initial edge> is the edge of H terminated by V
<final edge> is the edge of H initiated by V.

These triples will then be linked together to form the figure H. First of all, we will describe the algorithm used to form these ordered triples.

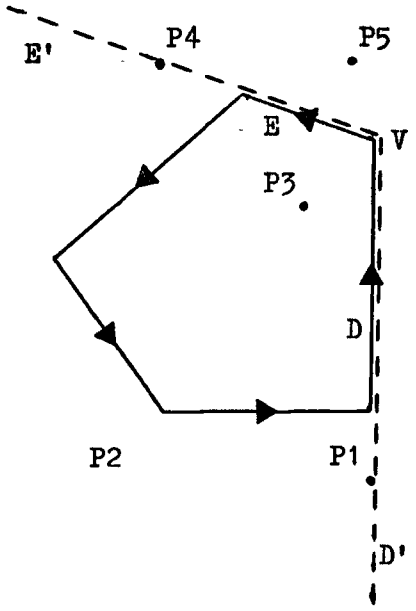


Diagram 6. Points P1, P2, and P3 are inside the angle formed by D and E at V, while points P4 and P5 are outside.

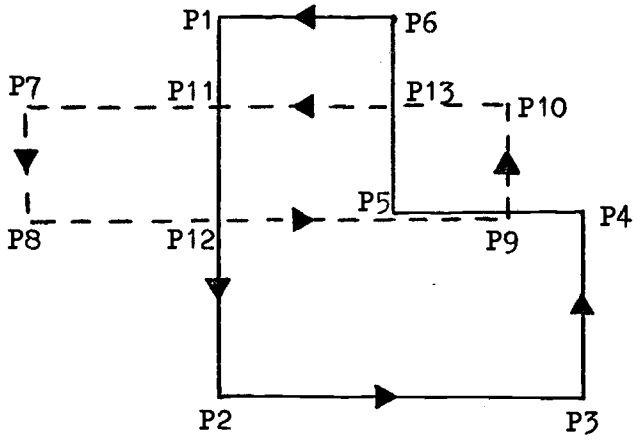


Diagram 7. P1 to P13 are the critical points of the pair of figures (F,G), where F and G are denoted by the solid and dashed lines respectively. P1 to P6 are critical points since they are vertices of F, P7 to P10 are critical points since they are vertices of G, and P11 to P13 are critical points because they are crossing points.

4.2.1 Algorithm For Forming Ordered Triples -

1. Find all critical points, i.e. all points which coincide with vertices of F or G, and all crossing points of the edges of F and G.

2. For each critical point P, construct its associated set S of vertices and edges.

3. Perform one of the following for each critical point P :

   1. S consists of only one vertex V:

      Say V is a vertex of F. If V is outside G, then do nothing. If V is inside G, then construct a vertex V which lies on P and form the triple (D,V,E) where D is the edge terminated by V, and E is the edge initiated by V. (see diagram 8)

   2. S consists of only two edges, D and E (which therefore must have a crossing point at P):

      Say D is an edge of F, and E is an edge of G. Construct the vertex V which lies on the crossing point of D and E. If the initial vertex of D is to the left of E, then form the triple (D,V,E). Otherwise form the triple (E,V,D). (see

diagram 9)

3. S consists of at least two vertices, V1, V2,...Vn, and no edges:

For each vertex Vi, let Di be its initial edge and Ei its terminal edge. Let Ui be the initial vertex of Di, and Wi be the terminal vertex of Ei. Look at all the Ui's. Find all the Ui's that are inside all the angles formed by the pairs of edges (D1,E1), (D2,E2),...(Dn,En). Say U1, U2,...Uk satisfy this requirement. If there are any pairs (Ui,Uj) such that Di and Dj lie on the same line, Di belonging to F and Dj belonging to G, then discard Uj. Say we are finally left with U1, U2,...Um. For each of these Ui's, find the Ek such that the interior angle formed by the pair (Di,Ek) is minimum. If there are two candidates (which might occur if two of the Ek's lie on the same line) then choose the Ek which belongs to F. Construct a vertex V which lies on P, and then form the triple (Di,V,Ek). (see diagram 10)

4. S consists of at least one vertex and one edge:

We transform this case to the previous one by replacing each edge E in S by a vertex V coinciding with P. We also form two edges, C and D, which are collinear and codirectional with E, where V is the terminal vertex of C and the initial

vertex of D. In effect, we regard each edge E in S as being a pair of edges joined at P by an angle of 180 degrees. We then proceed as in the previous case. If we produce any triples (C,V,D) where C or D have been constructed as above to replace some edge E, then we substitute E for C or D in the triple. Should both C and D have been constructed to replace the same edge E, we discard the triple altogether. (see diagram 11)

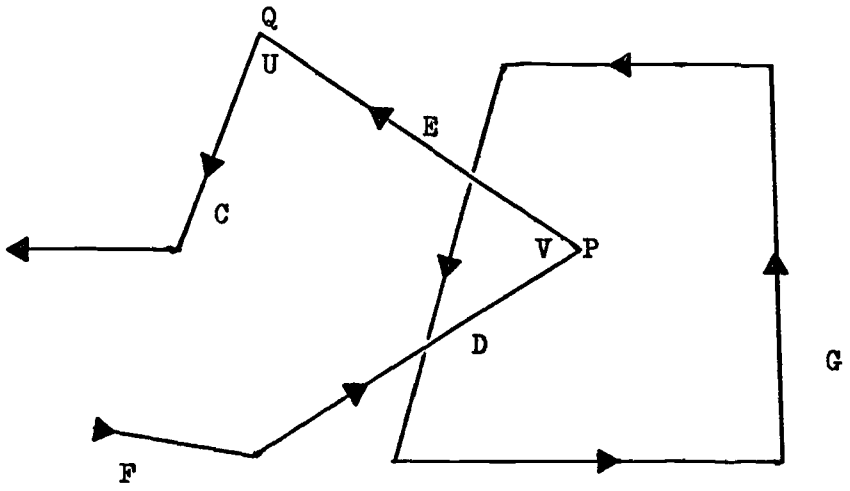


Diagram 8. Part of F and all of G are shown in this diagram. P is a critical point whose associated set S(P) consists only of the vertex V. Since V is inside G, we construct a vertex V′ which lies on P, and form the ordered triple (D,V′,E) where D and E are the edges terminated by and initiated by V respectively.

Q is a critical point whose associated set S(P) consists only of the vertex U. Since U is outside G, we do nothing.

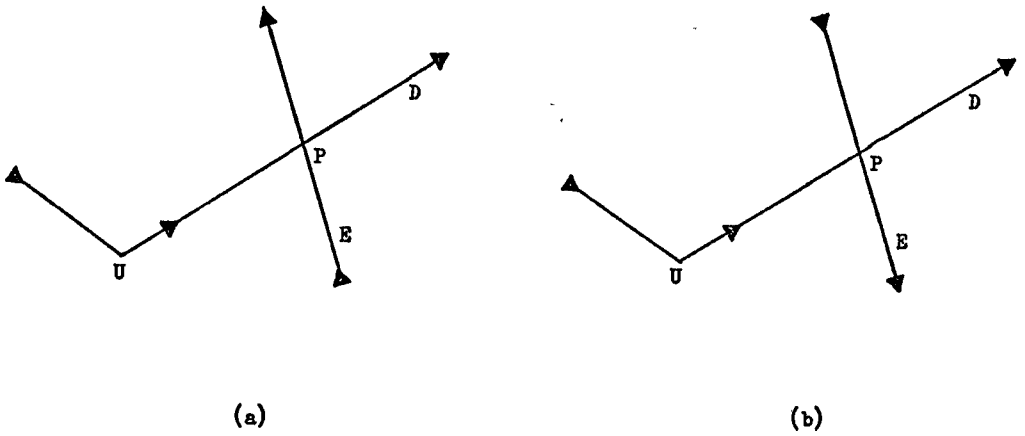


Diagram 9. In (a), P is a critical point whose associated set S(P) consists only of two edges D and E, which cross at P. D and E are edges of figures F and G respectively. We construct a vertex V which lies on P. U is the initial vertex of D, and since U lies to the left of E, we form the triple (D,V,E).

In (b), U lies to the right of E, hence we form the triple (E,V,D).

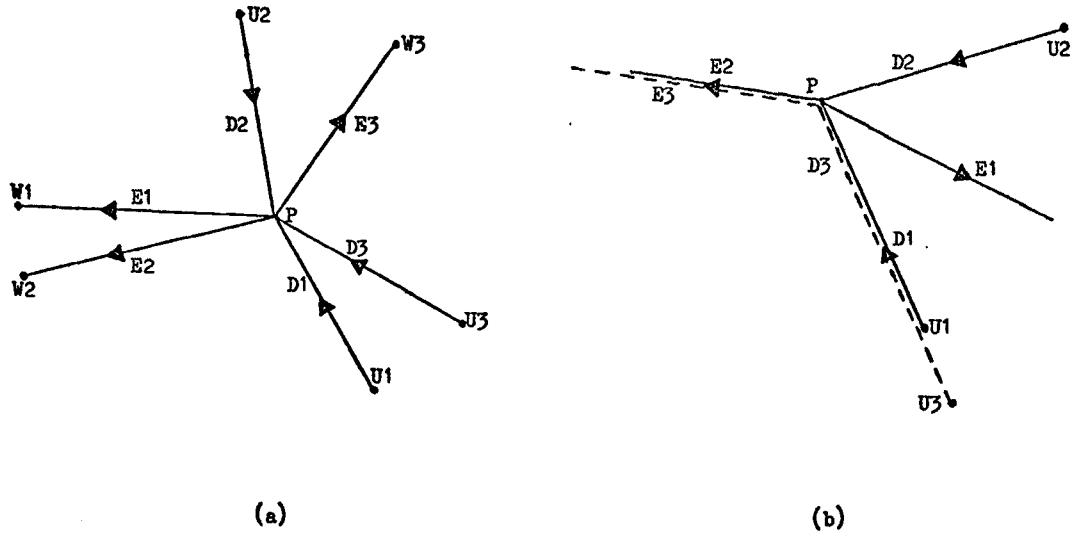
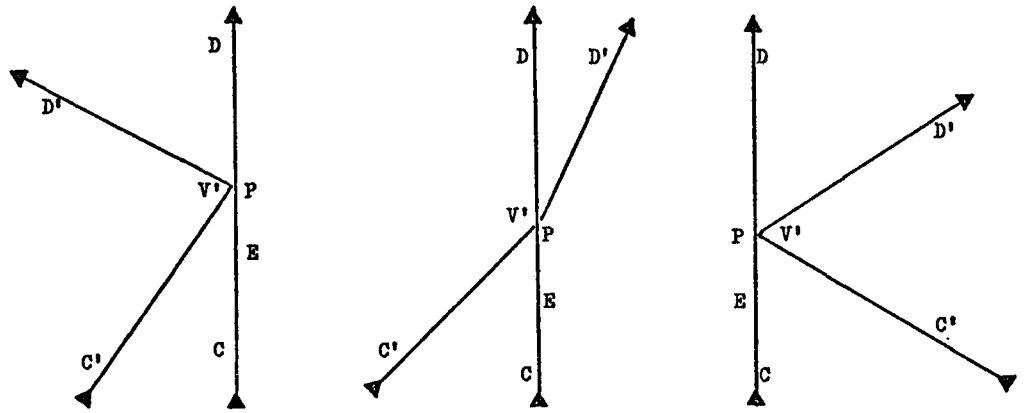


Diagram 10. In (a), we have a critical point P whose associated set consists of three vertices V1, V2, and V3, (not labelled in the figure), which lie on P. Of the Ui's, only U1 is inside all the angles formed by the pairs of edges (D1,E1), (D2,E2), and (D3,E3) (see diagram 6). Of the Ek's, E2 forms the minimum interior angle with D1. We construct a vertex V which coincides with P, and form the triple (D1,V,E2).

In (b), edges D1, D2, E1, and E2 belong to F, while edges D3 and E3 belong to G. Both U1 and U3 are inside the angles formed by the pairs of edges (D1,E1), (D2,E2), and (D3,E3). However, D1 and D3 lie on the same line, so we discard U3 in favour of U1 which belongs to F. Now both E2 and E3 form minimum interior angles with D1, so we discard E3 which belongs to G. Thus we form the triple (D1,V,E2).



(a) (b) (c)

Diagram 11. In these three diagrams, we have a critical point P whose associated set S consists of a vertex V′ and an edge E. We construct a vertex V which lies on P and associate with V the edges C and D which are collinear and codirectional with E. V is the terminal vertex of C and the initial vertex of D. We now proceed as in step 3.3. (Diagram 10.)

In (a), we produce the triple (C′,V,D′).

In (b), we produce the triple (C′,V,D). Since D was constructed to replace E, we substitute E for D and are left with the triple (C′,V,E).

In (c), we produce the triple (C,V,D). Since C and D were both constructed to replace the same edge E, we discard the triple altogether.

4.2.2 Algorithm For Linking Ordered Triples -

Assume that after applying the algorithm in the previous section, we are left with a set Q of ordered triples. Each of these triples represents a point where a directed curve of H, having coincided over some length with some edge of F or G, is now coinciding with some other edge. We must now link these triples together into ordered sets in such a way that each ordered set represents a distinct directed curve of H. This is done by applying the following algorithm.

1. If Q is empty, go to step 5.

2. Remove any triple T1=(D1,V1,E1) from Q, and assign 1 to the variable n.

3. Find all triples T=(D,V,E) in Q with D equal to En.

4. Do one of the following :

    1. If step 3 has produced only one T, then remove T from Q, increment n, and assign T to Tn. Go to step 3.

2. If step 3 has produced more than one T, then choose the T whose V is closest to Vn, but lies past Vn in relation to the direction of En (see diagram 12). Remove this T from Q, increment n, and assign T to Tn. Go to step 3.

3. If step 3 has produced no T, then En should equal D1, and the set (T1, T2,...Tn) represents one of the directed curves of H. Put it aside and go to step 1.

5. Say we have produced m ordered sets of triples R1, R2,...Rm. For each Ri = (Ti1, Ti2,...Tin(i)) form the ordered set Hi = (Vi1, Vi2,...Vin(i)). If there exists a Vij such that Vij-1, Vij, and Vij+1, where the second subscripts are taken mod n(i), lie on a straight line, then remove Vij from Hi. If we now regard each Hi as a permutation, then we have that H = (H1, H2,...Hm) is the intersection of the two figures F and G.

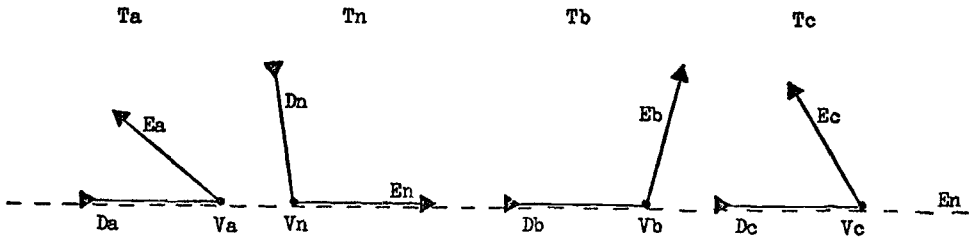


Diagram 12. In this diagram, we have three triples Ta, Tb, and Tc, whose first component is equal to En. Only Vb and Vc lie past Vn, and of these two, Vb is the closest to Vn. Hence Tb is removed from Q and is assigned to Tn+1.

4.3 An Example

Suppose we wish to find the intersection of the two figures, F and G, shown in diagram 13. Applying the first algorithm for forming ordered triples, we find that we have 14 critical points to consider. These points and their associated sets of vertices and edges are listed in table 1.

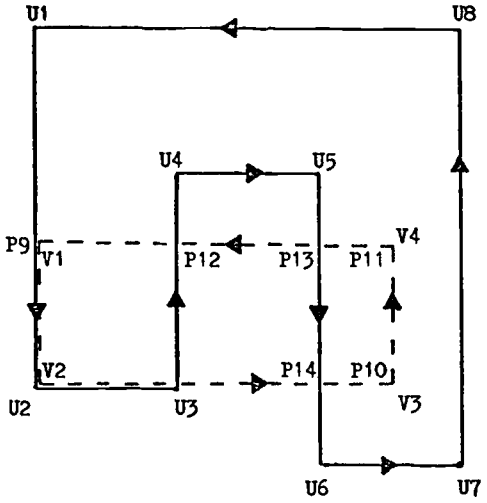
We now look at step 3 of the first algorithm. Step 3.1 applies to points P1, P4 to P8, P10, and P11. The associated set of P1 consists of the vertex U1 of F, which lies outside G; hence we do nothing. Similarly, consideration of points P4 to P8 does not result in the formation of any ordered triples. The associated set of P10 consists of the vertex V3 of G, which lies inside F. Hence we form the triple (E2,W10,E3) where W10 is a vertex constructed to lie on P10, and E2 and E3 are the initial and terminal edges respectively of V3. Similarly, consideration of point P11 results in the formation of the triple (E3,W11,E4).

Step 3.2 applies to points P12, P13, and P14, and results in the formation of the triples (D3,W12,E4), (E4,W13,D5), and (D5,W14,E2), where W12, W13, and W14, lie on P12, P13, and P14 respectively. Step 3.3 applies to point P2, resulting in the triple (D1,W2,D2). Step 3.4 applies to points P3 and P9, resulting in the triples (D2,W3,D3) and (E4,W9,D1).

We now have the set of ordered triples listed in table 1, and the algorithm for linking them can be applied. Suppose that in accordance with step 2, we remove (D1,W2,D2). Application of steps 3 and 4 results in the formation of the ordered set ((D1,W2,D2), (D2,W3,D3), (D3,W12,E4), (E4,W9,D1)). Note that step 4.2 had to be applied in order to choose the triple (E4,W9,D1) instead of (E4,W13,D5). We now go to step 1. As Q is not yet empty, we remove another triple, say (E2,W10,E3). This produces the ordered set ((E2,W10,E3), (E3,W11,E4), (E4,W13,D5), (D5,W14,E2)). Q is now empty so we go to step 5 and form the ordered sets H1 = (W2, W3, W12, W9) and H2 = (W10, W11, W13, W14). Thus the intersection of F and G is the set H consisting of the two permutations (W2 W3 W12 W9) and (W10 W11 W13 W14), as in diagram 13.

Point	Associated Vertices and Edges		Associated Triples
P1	U1		
P2	U2 V2		(D1 W2 D2)
P3	U3	E2	(D2 W3 D3)
P4	U4		
P5	U5		
P6	U6		
P7	U7		
P8	U8		
P9	V1	D1	(E4 W9 D1)
P10	V3		(E2 W10 E3)
P11	V4		(E3 W11 E4)
P12		D3 E4	(D3 W12 E4)
P13		D5 E4	(E4 W13 D5)
P14		D5 E2	(D5 W14 E2)

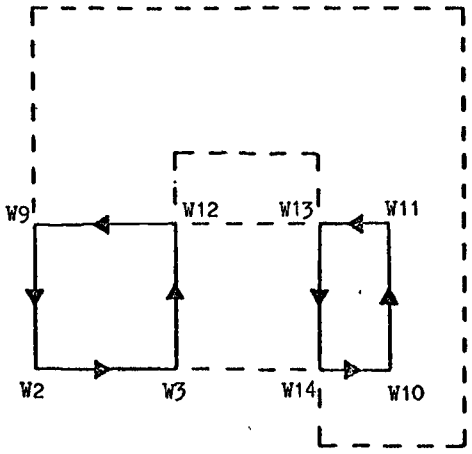
Table 1. The critical points of figures F and G from diagram 13 are listed here, along with their associated sets and ordered triples. Thus Q, the set of ordered triples, consists of all the entries in the last column of this table.



F={(U1 U2 U3 U4 U5 U6 U7 U8)}

G={(V1 V2 V3 V4)}

(a)



$$H=\left\{\begin{matrix}(W2\ \ W3\ \ W12\ W9\ )\\(W10\ W11\ W13\ W14)\end{matrix}\right\}$$

(b)

Diagram 13. In (a), figures F and G are illustrated by the solid and dashed lines respectively. Critical points P1 to P8 are not labelled, but coincide with vertices U1 to U8 respectively. Edges D1 to D8 of F (not labelled) are the edges initiated by U1 to U8 respectively. Similarly, edges E1 to E4 of G (not labelled) are the edges initiated by V1 to V4 respectively.

In (b), the intersection of F and G is illustrated by the solid lines.

5.0 GRASPING OBJECTS

We shall now describe how the material presented in section 4 can be applied to the problem of grasping objects in a cluttered environment. Suppose we have a specific body B which we wish to grasp. According to the restrictions of section 2, we grasp bodies by parallel and opposite faces, i.e. faces whose normals have opposite directions and point away from the opposite face. Furthermore, suppose faces F1 and F2 satisfy the preceding condition. Let us project F2 onto the plane which contains F1. Let us refer to this projection as F2′. Then if we are to grasp body B by faces F1 and F2, it is necessary that the intersection of F1 and F2′ be nonempty. All the pairs of faces which satisfy the above two conditions are potentially suitable for grasping.

This list can be shortened by further considerations. First of all, the distance between the faces cannot be greater than the maximum separation of the hands. Also, suppose that after grasping body B we wish to manipulate it so that one of its faces, F say, is against a wall. Then we should not grasp B by any pair of faces of which F is a member. The centre of gravity of B can also be taken into account in determining which pair of faces might be more suitable for grasping than others. The last two considerations have not been incorporated into the system.

In any case, assume that we have been able to find some pairs of faces of B which meet our requirements for grasping. The major problem now is to examine these pairs one by one until we find a trajectory for grasping one of these pairs which does not result in a collision between the robot and any body in the robot's world. This will be dealt with in detail in the following sections.

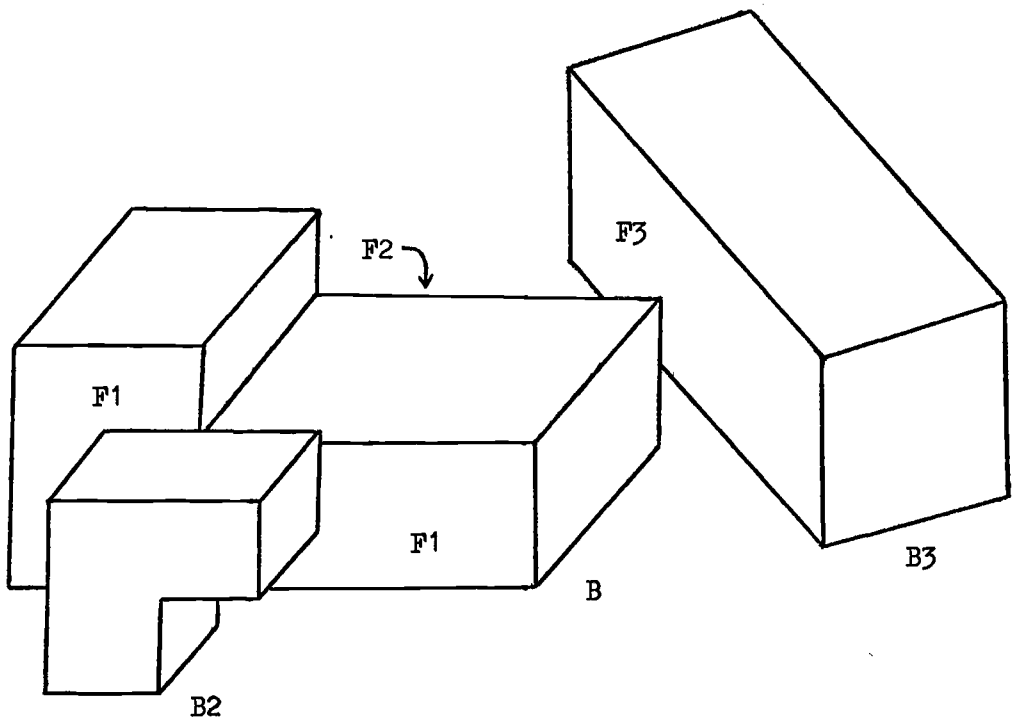


Diagram 14. Consider the above situation where the robot's world consists of the three bodies B, B2, and B3. Suppose we wish to grasp B by faces F1 and F2. We must find a trajectory of the hands which will not result in a collision with bodies B2 or B3.

5.1 Projecting Obstructions Onto A Plane

The following material will be illustrated by the example in diagram 14. Assume that we have a set of bodies in the robot's world, each represented as in section 3. Assume that we wish to grasp a certain body B, and that we have found a list of pairs of faces which are potentially suitable for grasping, according to the requirements outlined in the preceding section. Let us choose some pair of faces, say (F1, F2), from this list. We wish to find some trajectory which will allow the robot to grasp body B by these faces without colliding into any body in the robot's world.

According to section 2, our trajectories will be of the form shown in diagram 1. Let us consider the final part of the trajectory, i.e. when the robot hands are moved toward the body until contact is established with the two opposite faces. This final part of the trajectory determines the thickness, d, of each of the cuboids in diagram 1. Suppose we choose some point P on one of the faces and draw a normal through P, outward from the face. Then if this normal intersects some body at a distance less than d from P, it will be impossible to grasp the body by any surface region which includes P. Thus a necessary (but not sufficient) condition for grasping a body at a certain surface region is that no point in that region should be within a distance d of another body in the robot's world. Our first step is therefore to find all the points on faces F1 and F2 of B which do not satisfy this condition.

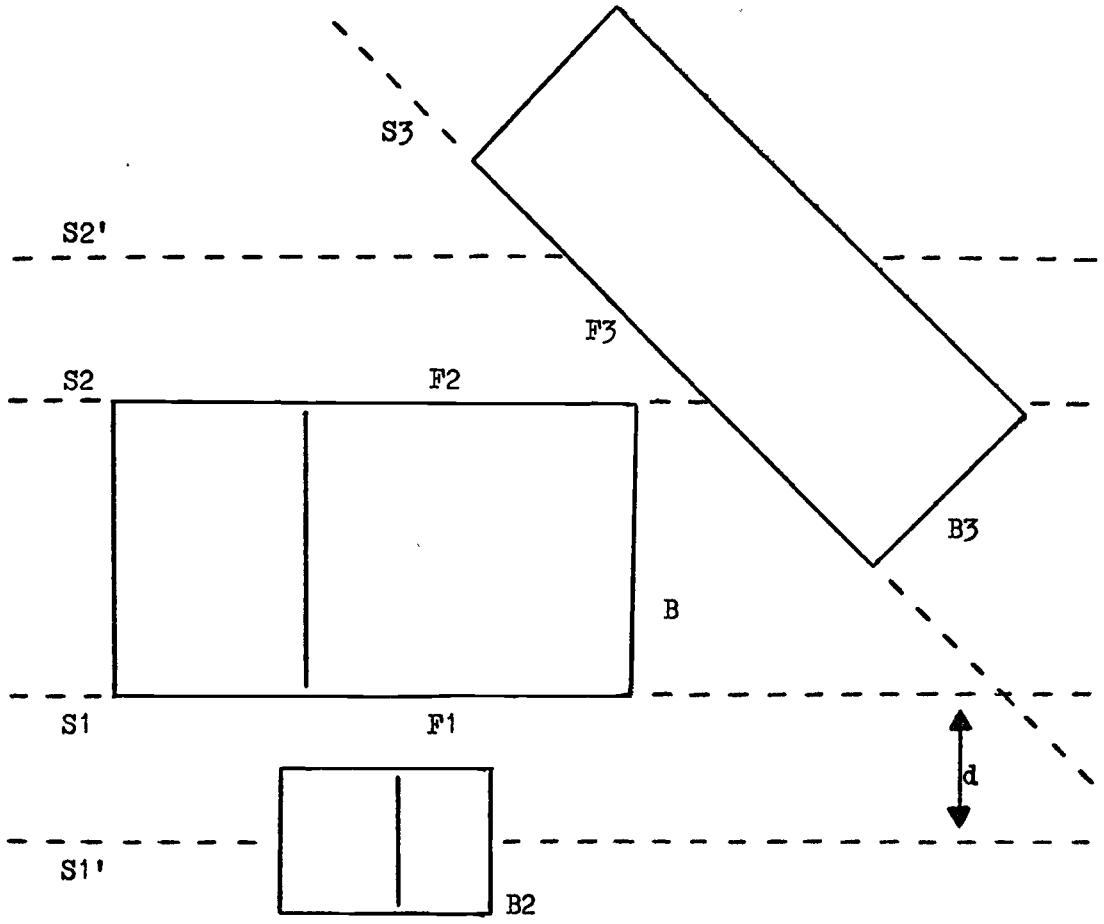


Diagram 15. This is a bird's eye view of the situation depicted in diagram 14. S1 and S2 are the planes which contain faces F1 and F2 respectively, while planes S1' and S2' are parallel to and at a distance d from these faces. Any portion of bodies which lie between S1 and S1' or S2 and S2', will be an obstacle to the robot.

Let us refer to the plane in which F1 lies as S1, and the plane in which F2 lies as S2. We also consider planes S1′ and S2′ which are parallel to and at a distance d on the outside of planes S1 and S2 respectively (see diagram 15). We are interested in points within bodies, which lie between planes S1 and S1′ or S2 and S2′. The projection of all such points onto S1 or S2 represents a region on S1 and S2 which is inaccessible to the robot. We must therefore determine the projection of all such points.

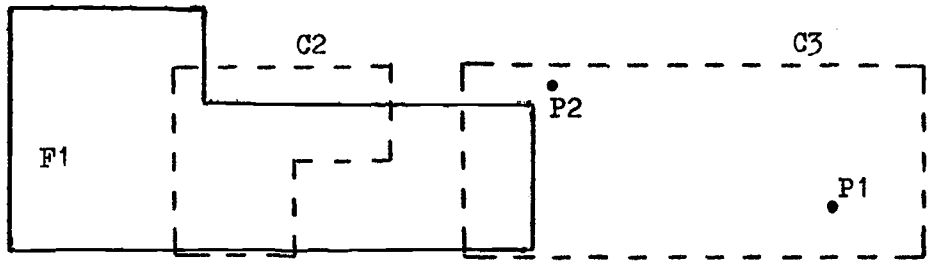


Diagram 16. This diagram is in the plane S1. F1 is denoted by the solid line. C2 represents the union of all cross-sections of B2 parallel to and between S1 and S1′. C3 represents the union of all cross-sections of B3 parallel to and between S2 and S2′. Any point in C2 or C3 is within a distance d of an obstacle. Thus any attempt to move the robot hands into this region will result in a collision. The union of C2 and C3 is referred to as U.

This can be done by taking all the cross-sections parallel to S1, of the portions of bodies which lie between S1 and S1′ or S2 and S2′, and then projecting all these cross-sections onto S1 (see diagram 16). Note that a body which lies between S1 and S1′ results not only in a region of S1 which is inaccessible to the robot, but an identical region of S2 as well. This is because the two hands of the robot move together and thus if a region on one face is inaccessible to the robot, the corresponding region on the opposite face is inaccessible as well. Therefore we can project all obstructions onto one plane, say S1, rather than projecting those between S1 and S1′ onto S1, and those between S2 and S2′ onto S2.

Instead of calculating the projections of all the relevant cross-sections, we do the following. First of all we project onto S1 all the portions of faces of bodies, which lie between S1 and S1′ or S2 and S2′. We then project onto S1 the intersections of S1 and S2 with all bodies. It can be easily seen that both methods result in same region of projections on S1. Clearly the region created by the first method contains the region created by the second. The converse is also true: Take any point P in the region created by the first method. Point P corresponds to the projection onto S1 of some point P′ in a cross-section of some body B′. Let us draw a normal to S1 which passes through P′. Assume P′ lies between S2 and S2′. We examine the point P′′ where the normal intersects S2. If P′′ lies inside B′, then P′′ is a member of the intersection of B′ with S2, and hence P is contained in the projection of this intersection onto S1. On the other hand, if P′′ lies outside B′,

then the line P'P'' must pass through a face of B' at some point between S2 and S2', and hence the projection of this face onto S1 contains P (see diagram 17). Thus the region of projections created by the second method contains the region of projections created by the first; hence the two regions are identical.

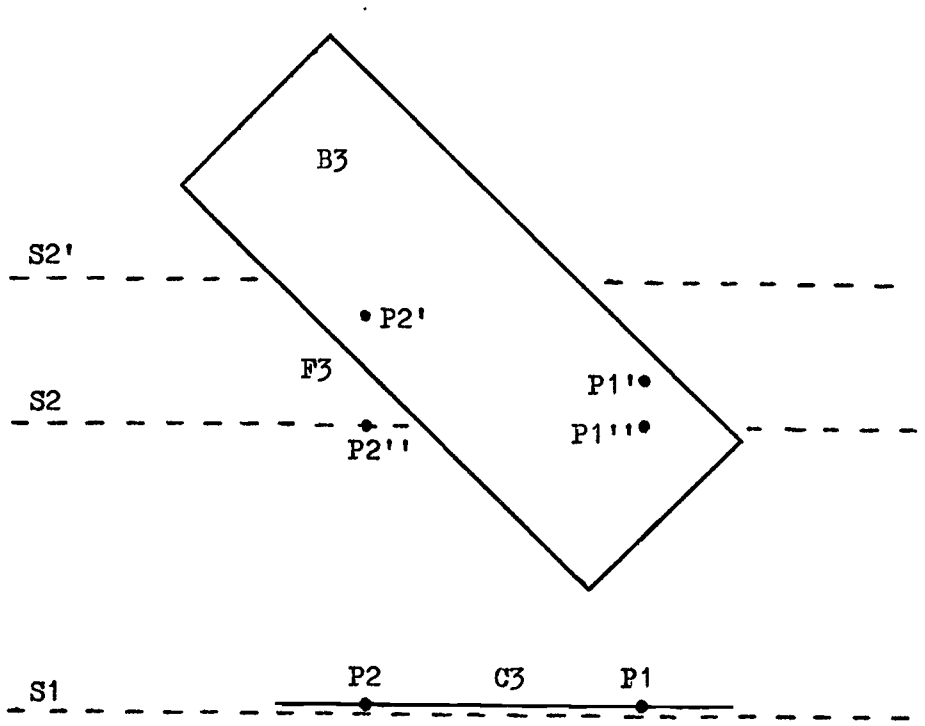


Diagram 17. Consider the two points P1 and P2 in region C3 of diagram 16. They correspond to the projection of some points P1' and P2' in the portion of body B3 which lies between S2 and S2'. P1'' and P2'' are the projections onto S2 of P1' and P2' respectively.

Since P1'' lies in B3, the projection onto S1 of the intersection of S2 with B3 will contain P1.

Since P2'' lies outside B3, the line P2'P2'' must intersect some face of B3 (in this case F3) at some point between S2 and S2'. Thus the projection onto S1 of the portion of F3 which lies between S2 and S2' will contain P2.

In practice, it is sufficient to consider only those bodies which are close enough to body B to be a possible obstruction (including body B itself). For this purpose, each body has a radius which is defined to be the maximum distance from the local origin of the body axes to any point on the body. Suppose B has radius r and local origin O, and some body B3 has radius r′ and local origin O′. Then B3 cannot be an obstruction if the distance between O and O′ is greater than the sum of r, r′, and d. If B3 cannot be rejected by this test, we see whether B3 lies totally on the side of S1′ or S2′ opposite from faces F1 and F2 of B. This will be true if O′ is on the outside of and at a distance of at least r′ from either S1′ or S2′. If this test fails as well, we must consider B3 in more detail. This involves examining each of the faces of B3.

Let us choose some face F3 of B3. We wish to determine the portion of F3 which lies between S1 and S1′ or S2 and S2′. First of all, we calculate the angle between F3 and S1. If this is close to zero, i.e. if F3 is almost parallel to S1, then we check whether the local origin of F3 is within the two pairs of parallel planes. Since F3 is almost parallel to these planes, we can assume that F3 lies within the restricted volume if and only if the local origin of F3 lies within it. Thus if the local origin of F3 lies between the parallel planes, we project all of F3 onto S1; otherwise we conclude that F3 is not an obstruction and ignore it.

In the more general case where F3 is not almost parallel to S1, we determine the region of F3 which lies between the two pairs of parallel planes (S1, S1′) and (S2, S2′) as follows. First of all, we find the intersection of planes S1, S1′, S2, and S2′, with the plane S3 in which F3 lies. Let us denote these lines of intersection by L1, L1′, L2, and L2′ respectively. We then calculate the region of F3 lying between the pairs of lines (L1, L1′) and L2, L2′). This is done by joining each pair of lines at a notionally large distance from F3 so as to form two rectangles (see diagram 18). We then use the method of section 4 to intersect these two rectangles with F3, which gives us the desired region. This region is then projected onto S1. (In the case of F3 being perpendicular to S1, we do not calculate a projection, as it would merely be a straight line rather than a region.) The intersections of S1 and S2 with body B3 are also calculated and projected onto S1.

Once we have gone through the above process with each of the bodies in the robot's world, we take the union U, of all the projections we have made onto S1. Thus U represents a region on planes S1 and S2 (in the vicinity of B) which is inaccessible to the robot (diagram 16).

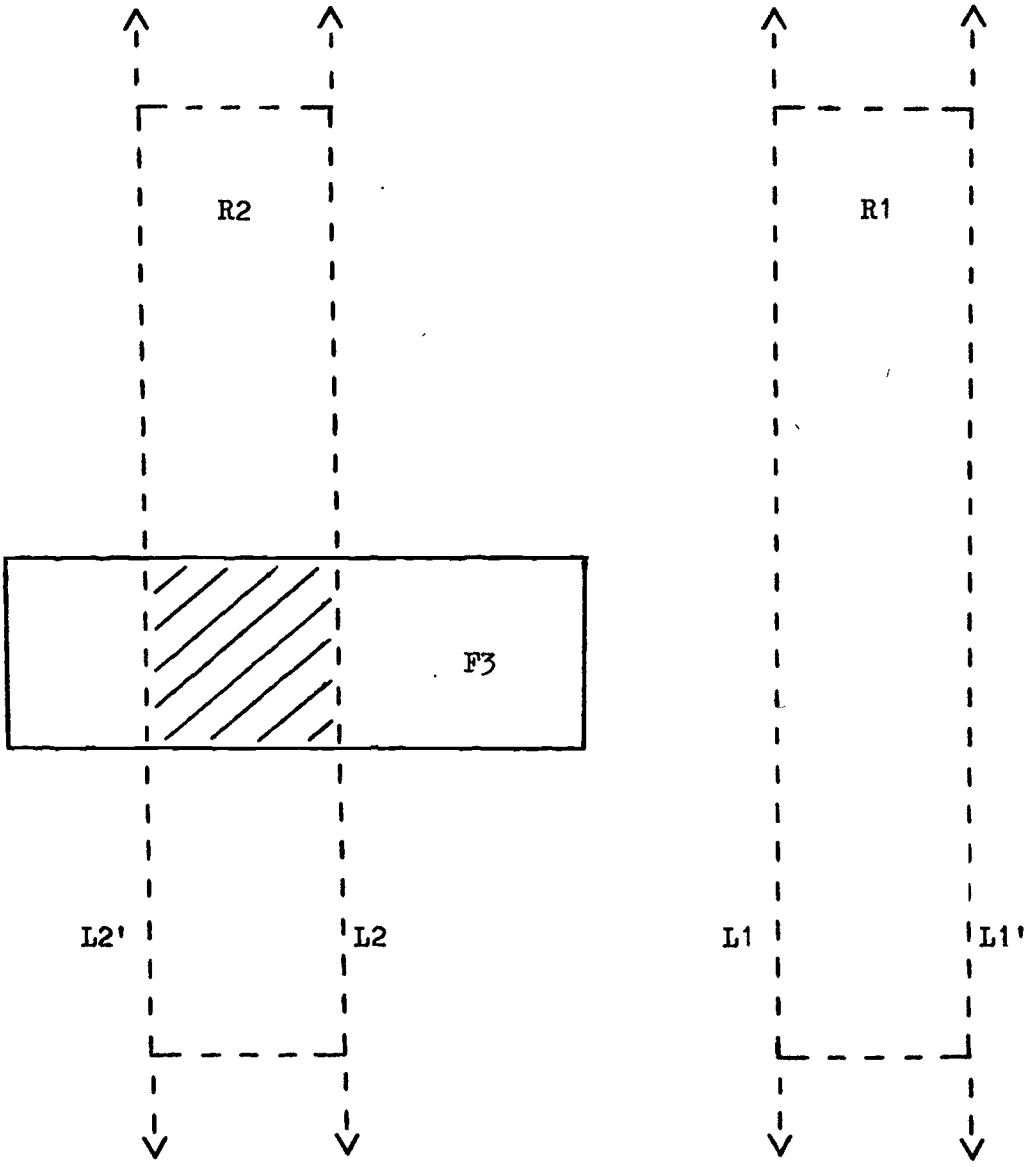


Diagram 18. This diagram is in the plane S3. We consider face F3 of body B3 (diagram 15). L1, L1′, L2, and L2′, represent the intersections of S3 with S1, S1′, S2, and S2′ respectively. We join these lines at a notionally large distance from F3 so as to form the two rectangles R1 and R2. The intersection of R1 and R2 with F3, represented by the hashed region, is the portion of F3 lying between the pairs of planes (S1, S1′) and (S2, S2′). This portion is then projected onto S1.

If we do this for each face of B3, and project onto S1 the intersections of B3 with S1 and S2, we will obtain the region C3 in diagram 16.

5.2 Computing The Inaccessible Region

In the previous section we have been able to obtain a region U of planes S1 and S2 which is inaccessible to the robot. However, this is only part of the total region of planes S1 and S2 which is inaccessible to the robot. Assume that the hands are coming vertically down to grasp an object. Then as in diagram 19, it can be seen that the region U "casts a shadow" which is inaccessible as well. This shadow is cast in the direction of the trajectory of the robot hands and can be easily calculated given the region U. As mentioned in section 2, one tends to grasp objects from above, i.e. using a downward trajectory. We shall only consider trajectories of this type. (Indeed, for the Edinburgh University robot, the volume swept by a downward trajectory to grasp an object at a certain point is contained in the volume swept by any other grasping trajectory. Therefore in this case, a vertical trajectory is uniformly the most efficient for grasping objects.) We calculate the region U′, which corresponds to casting a shadow vertically downward on U (see diagram 19).

The other point to consider is that the robot hands have a certain width w (corresponding to the width of the cuboids in diagram 1), which will prevent the hands from passing through a narrow gap. If we represent the position of each robot hand by the midpoint of the bottom inner edge of the hand (diagram 20), then we must allow a clearance of w/2 to the left and right of the position of each hand. This corresponds to extending the inaccessible region

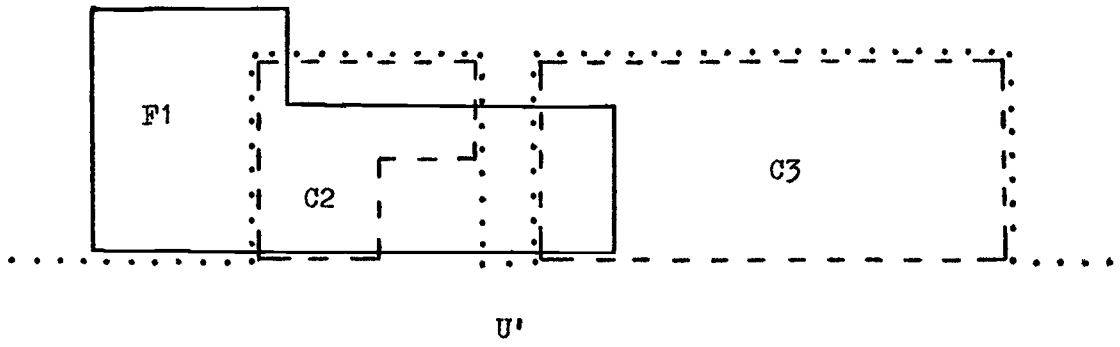


Diagram 19. For the situation depicted in diagram 14, we have been able to obtain the inaccessible region U (consisting of the union of C2 and C3) shown in diagram 16. Assuming the hands are moving vertically downward, any point below an obstruction is inaccessible as well. Thus our inaccessible region can be enlarged to include all such points, represented here by the region U' below the dotted line.

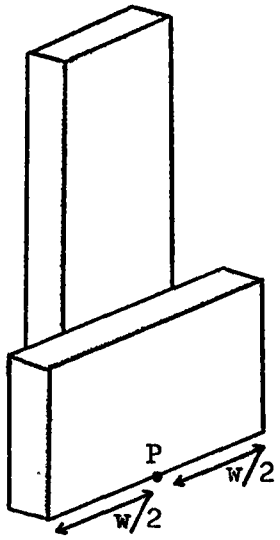


Diagram 20. This shows one ot the robot hands from diagram 1. The position of the hand is denoted by the position of the point P. Assuming the hand has width w, we must allow a clearance of w/2 to either side of P when calculating a trajectory.

U′ leftward and rightward by an amount w/2 (see diagram 21). This results in region U′′, the total region of S1 and S2 which is inaccessible to the robot; i.e. there exists a vertical trajectory to any point of S1 or S2 which does not lie in U′′.

It is now only necessary to find which of these points lie within the intersection of faces F1 and F2 of body B. We do this by projecting F2 onto S1 (the plane in which F1 lies) to obtain the projection F2′. We find the intersection I of F1 and F2′, which corresponds to the overlap of the two faces. As mentioned in section 5.0, we have already ensured that I is nonempty. We then take the complement of U′′ and intersect it with I. The resulting region R corresponds to all the points on faces F1 and F2 to which a vertical trajectory exists (see diagram 22). (The complement of U′′ is computed simply by reversing the order of each permutation of points in the figure U′′. cf. section 3.1) If R is empty, it would be necessary to examine another pair of faces of B. Assuming this is not the case, any point in R is suitable for grasping; i.e. for any point in R, there exists a vertical trajectory of the type described in diagram 1 which will result in the robot grasping body B by faces F1 and F2 without a collision.

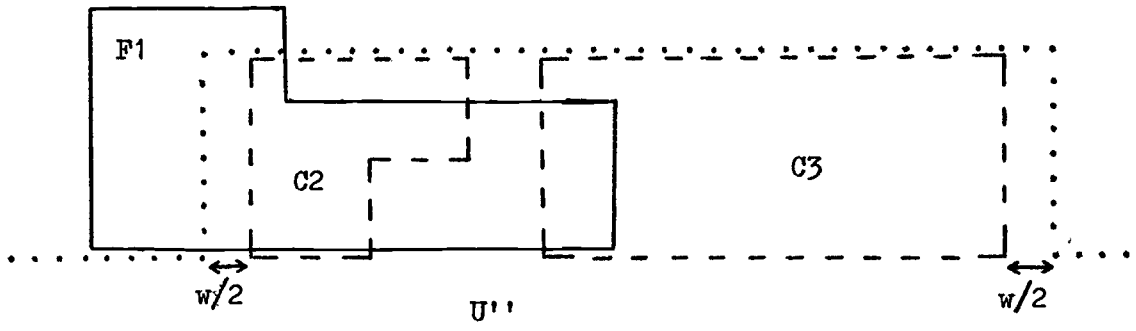


Diagram 21. We proceed from diagram 19. Taking into account the width w of the robot hands, we expand the boundary of U' to the left and right by the amount w/2. We thus obtain the inaccessible region U'', represented by the region below the dotted line. The robot hands can reach any position above the dotted line (via a vertical trajectory) without a collision. Note that the region between C2 and C3 is inaccessible, as the distance between their top edges is less than w.

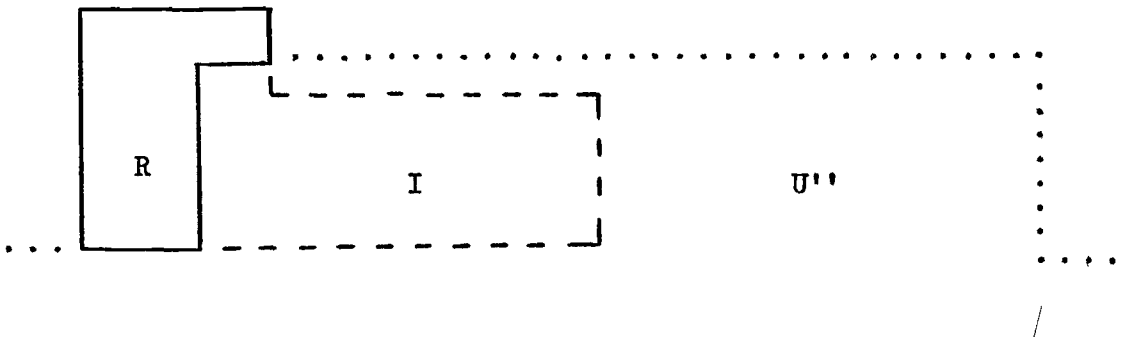


Diagram 22. Continuing from diagram 21, we compute the complement of U'', and the overlap I of faces F1 and F2 (which in this case is equivalent to F1). We intersect I with the complement of U'', to obtain the region R denoted by the solid line.

6.0 A WORKING SYSTEM

The preceding material has been made into a working system using the Edinburgh University robot and DEC-10 computer. The implementation is totally in POP-2 on the DEC-10, except for the programs which supervise the actual movements of the robot. These are written in POPLET (a restricted version of POP-2) and reside on a Honeywell 316 which is dedicated to the robot. Communication between the DEC-10 and the Honeywell is made via a high-speed line.

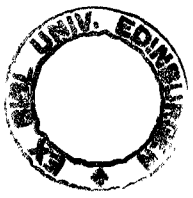
The shape and position of each object in the robot's world is input to the system. In an environment consisting of about four objects in awkward positions, the system is able to compute and perform a trajectory to grasp any specified object in the order of 30 seconds CPU time. No attempt has yet been made to optimize the coding, and should this be done, a tenfold increase in efficiency could be reasonably expected. Also, most of the algorithms are parallel in nature and a significant decrese in processing time could be achieved by using more than one processor.

This system would be able to be linked up with a visual recognition system now being developed at Edinburgh, so that the position of bodies in the robot's world can be determined visually rather than being explicitly input to the system.

7.0 CONCLUSION

This work takes a much less restricted approach to the problem of grasping objects than do previous systems such as TC-COPY [Winston, 1971], AL [Finkel et al., 1974], and VAP [Ambler et al., 1975], as described in the introduction. This is a significant advance, since in the field of robotics and automatic assembly, one wishes to dispense with the job of specifying trajectories to perform certain tasks - a job which becomes cumbersome if one has to worry about collisions with nearby objects. Performing tasks solely by describing changes in the robot's environment, allowing the system to calculate the necessary trajectories to achieve these changes, is an objective one would like to obtain. Since a large proportion of these trajectories would involve the grasping of objects, this work goes a substantial way toward achieving this objective. Merely by specifying the object to be grasped, the system is capable of computing a set of trajectories which avoid collisions with nearby objects. It is able to deal with an arbitrary arrangement of objects with planar faces.

The representation of bodies in terms of their surfaces as opposed to their space occupancy has proved to be useful. This is because all physical interactions between bodies occur at their surfaces. Enclosing each body in a sphere, however, allows us to quickly find those bodies which can be ignored because of their distance from the object to be grasped.



One would like to extend this system in two ways. First of all, it would be desirable to be able to deal with objects of arbitrary shape. As the approach taken in this paper depends heavily on the fact that surfaces are planar, the best solution is to approximate curved surfaces by planar faces. Secondly, one would like to be able to find trajectories of more complex-shaped or jointed bodies through a cluttered environment. Specifically, having grasped an object, one would like to be able to compute a trajectory to move the object to another position, taking into consideration the joints on the robot. This is an extremely difficult problem to which no total solution is apparent. The best method of approach is probably to assume that no obstacles are present above a certain height from the working area. Thus the robot could move freely above this height and we would need only concern ourselves with trajectories which raise or lower objects from the working area. This is certainly a feasible extension of the present system.

As mentioned in the preceding section, this system would become much more versatile when linked to a visual recognition system, as this would avoid the difficulty of having to explicitly input the position of each object in the robot's environment. Of course visual recognition, which tends to be quite time-consuming, would only be needed initially, as the system would be able to keep track of any subsequent changes to the robot's world.

Ultimately, this work could be applied to an automatic assembly system where given a description of the shape of each component, a specification of the physical relations which are to hold between components in each step of the assembly, and the order in which these steps should be carried out, the system would be able to analyze a visual scene containing the components and compute suitable trajectories to perform the assembly.

8.0 REFERENCES

1. Ambler, A. P., Barrow, H. G., Brown, C. M., Burstall, R. M., and Popplestone, R. J., "A Versatile System for Computer-Controlled Assembly", Artificial Intelligence, Vol. 6, pp. 129-156, 1975.

2. Ambler, A. P., and Popplestone, R. J., "Inferring the Positions of Bodies from Specified Spatial Relationships", Artificial Intelligence, Vol. 6, pp. 157-174, 1975.

3. Braid, I. C., "The Synthesis of Solids Bounded by Many Faces", Communications of the ACM, Vol. 18, No. 4, pp. 209-216, April, 1975.

4. Fikes, R. E., and Nilsson, N. J., "Strips : A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, Vol. 2, pp. 189-208, 1971.

5. Finkel, R., Taylor, R., Bolles, R., Paul, R., Feldman, J., "AL, a Programming System for Automation", Stanford University Artificial Intelligence Memo AIM-243, November, 1974.

6. Sacerdoti, E. D., "The Nonlinear Nature of Plans", Stanford University Artificial Intelligence Group Technical Note 101, January, 1975.

7. Sussman, G. J., "A Computational Model of Skill Aquisition", Tech. Note AI TR-297, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, August, 1973.

8. Winston, P. K., "Wandering About the Top of the Robot", Vision Flash 15, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, July, 1971.